Subject: : AmigaOS4
Topic: : LTO in AmigaOS4 gcc
Re: LTO in AmigaOS4 gcc
Author: : kas1e
Date: : 2019/2/16 18:34:44
URL:

@Andy
Quote:

> LTO isn't really going to do too much until you have multiple object files which do something meaningful, that can be benchmarked.

Yeah, will try tomorrow to build some fat by objects game with fps counter

Quote:

> Your test case is stupidly simple (no insult intended) it does nothing, calling my_malloc() has no efffect (result is unused) so it's been 'inlined then removed' by the -03 optimisation, then I expect the LTO optimsation has noticed that my_malloc() ends up not being called and removed that too. Normal optimsiation wouldn't remove the function as there would be no way to know if it was called from another module till link time.

Sure test case suck, but that point out that -flto is smart enough to make some optimisation even on single object file, just to even have 100bytes less exe.

Quote:

> If you split out your my_malloc() into another src file so that there are two objects file before linking you might see a bigger difference as the non LTO case would not be able to inline / remove it as it coudn't know if there were side effects to the function call. LTO (I would guess) would end up somewhat similar to the current result, depending on just what link time optimisations are possible.

Tried that:

```
 $ cat my_malloc.c

#include <stdio.h>
#include <strings.h>

int *ptr;

void* my_malloc(size_t nbytes)
{
                if (nbytes == 0)
                 return NULL;
             /* more code here */
             return ptr;
}


$ cat main.c

#include <stdio.h>
#include <strings.h>

#include <stdio.h>
#include <strings.h>

void* my_malloc(size_t nbytes);

int main()
{

        my_malloc(100);
        my_malloc(100);
        my_malloc(100);
        my_malloc(100);

}
```

And then:

```
 ppc-amigaos-gcc -flto -O3 -c my_malloc.c -o my_malloc_with_lto.o
ppc-amigaos-gcc -flto -O3 -c main.c -o main_with_lto.o
ppc-amigaos-gcc -flto -O3 main_with_lto.o my_malloc_with_lto.o -o test_with_lto

ppc-amigaos-gcc -O3 -c my_malloc.c -o my_malloc_no_lto.o
ppc-amigaos-gcc -O3 -c main.c -o main_no_lto.o
```

```
ppc-amigaos-gcc -O3 main_no_lto.o my_malloc_no_lto.o -o test_no_lto
```

And now, binary where no lto used, start to be bigger than when it single object version: 6247 vs 6176, while lto version in both cases be it 1 object or two, are 6107. So in case with single object version it better on 69 bytes, and in case with 2 objects, its better on 140 bytes (seems like you expect).